- Justificación de:
    - Los elementos del modelo que fueron necesarios persistir
    - Cómo se resolvieron los impedance mismatches.

Aclaración: no olvidar las estrategias de mapeo de herencia, si fueran necesarias. o Las estructuras de datos que se desnormalizaron, o que deberían estarlo.

Elementos Persistidos:

- actividad
- contactos
- dato_Contacto
- estacion
- factor_emision
- huella_carbono
- línea
- localidad
- medio_no_motorizado
- métrica_organizacion
- miembro
- municipio
- organización
- país
- provincia
- punto_geografico
- sector
- sector_territorial
- servicio_contratado
- solicitud_vinculacion
- tramo
- transporte_publico
- trayecto
- trayecto_pendiente
- trayecto_tramo
- usuario
- vehículo_particular

Resolución De impedance mismatches que tuvimos a lo largo del mapeo de nuestras entidades:

Nos encontramos con SectorTerritorial que es una clase abstracta de las cuales se desprende SectorMunicipal y SectorProvincial que implementan a la clase abstracta antes mencionada. Utilizamos la estrategia de inherital type single table realizando una única tabla (“SingleTable”) y para poder identificar cual es provincial y cual territorial la discriminamos con un String que va a ser una columna de la tabla. Por lo tanto, de esta manera teniendo una sola tabla de sector territorial tenemos todos los datos tanto provinciales como municipales discriminados por un String en una columna.

También nos encontramos con un problema en el cual tuvimos que desarrollar nuevo código para solucionarlo.

Dentro de la entidad HuellaDeCarbono que es una tabla, hay un atributo que es Unidad, nos dice la unidad en que esta expresada la huella de carbono. El problema es que esta unidad es una clase y nosotros no consideramos que sea necesario que la tabla HuellaDeCarbono tenga una columna con un id a otra tabla para poder ver SOLAMENTE la unidad. Por lo tanto, lo que agregamos fue un método en cada entidad que implementa Unidad (vale aclarar que la Unidad es una clase abstracta y hay varios tipos de unidad que la implementan) que nos devuelve un Sting que es el nombre de la unidad (nombre de la clase) que hace referencia a la unidad de la huella de carbono.

De esta manera a nuestro ORM le decimo que en unidad de la HuellaDeCarbono realice una columna String con el valor de la unidad. ¿Como lo hacemos? Realizamos un Coverter (@Converter) donde swichea la clase con un String y sabe cómo transformarla en Sting a la clase unidad.

Otra situación que tuvimos fue a la hora de crear la tabla Actividad, la actividad es una clase abstracta implementada por distintas activades (valga la redundancia). Elegimos tener todos los atributos juntos ya que no eran muchos y además cuando necesito la actividad casi el total de los casos voy a necesitar todos estos atributos, y es mucho más eficiente que me traiga todo de una a que tenga que traerme varias tablas. ¿Pero qué pasa? Nosotros juntamos todos los datos, pero uno de esos datos no era primitivo, era una clase Consumo, entonces en esa clase íbamos a tener el id_consumo hacia otra tabla. Por lo tanto, para solucionar esto y que la consulta a la base sea más eficiente y sabiendo que cuando le peguemos íbamos a querer toda esta información que está en la tabla y nunca solo ver actividad sin su consumo, embediamos consumo, a diferencia de la situación anterior donde realizamos un @Converter, en esta ocasión embediamos haciendo que todos sus atributos que tenía para persistir en otra tabla diferente se guarden en la tabla actividad. Logrando tener una tabla actividad con todos los necesario, pocos nulos y muy eficiente para nuestro contexto.

Además, una situación que se nos presento fue que la entidad MetricaDeOrganizacion que es la fila del Excel, decidimos guardarla como una tabla. Hasta ahí todo perfecto. Estas métricas tienen una activad que es un id actividad que línkea a la otra tabla como explicamos anteriormente. Lo cual nos resulta coherente que este en otra tabla ya que no siempre que necesite una metricaDeOrganizacion voy a querer la tabla de actividad. Pero PeriodoImputacion era una clase, lo embediamos también para tener dos columnas en metricaDeOrganizacion de local date y Sting periodicidad. Y de esa, manera se adecue mejor a nuestro contexto.

Para finalizar por el momento no consideramos que necesitemos desnormalizar estructuras de datos. Ya que los casos donde podía ser útil embediamos y no se termino desnormalizando los datos ya que se almacenan en un solo lugar.